دوال الاعادة الذاتية :

- تعني الاستدعاء للدالة لنفسها أكثر من مرة
نضع شرط التوقف باستخدام عبارة if الشرطية .

تمرين : برنامج مجموع الأعداد من (1) الى (n)

```
#include <iostream.h>
int fact (int x);
Void main()
{
int n,f;
cin>> n;
y=fact(n);   cout<<y;
}
int fact (int x)
{
if (x<=1)
return (1);
else
return (x + fact(x-1));
}
```

تمرين : اكتب برنامج باستخدام دوال الاعادة الذاتية لحساب a^b

```
#include <iostream.h>
int pw (int a, int b);
Void main()
{
int x,y,z;
cin>>x>>y;
```

```
Z = pw(x,y);   cout<<"\n Z ="<<Z;
}
int pw(int a , int b)
{
in(b==0)
return (1);
else
return (a* pw(a,b-1));
}
```

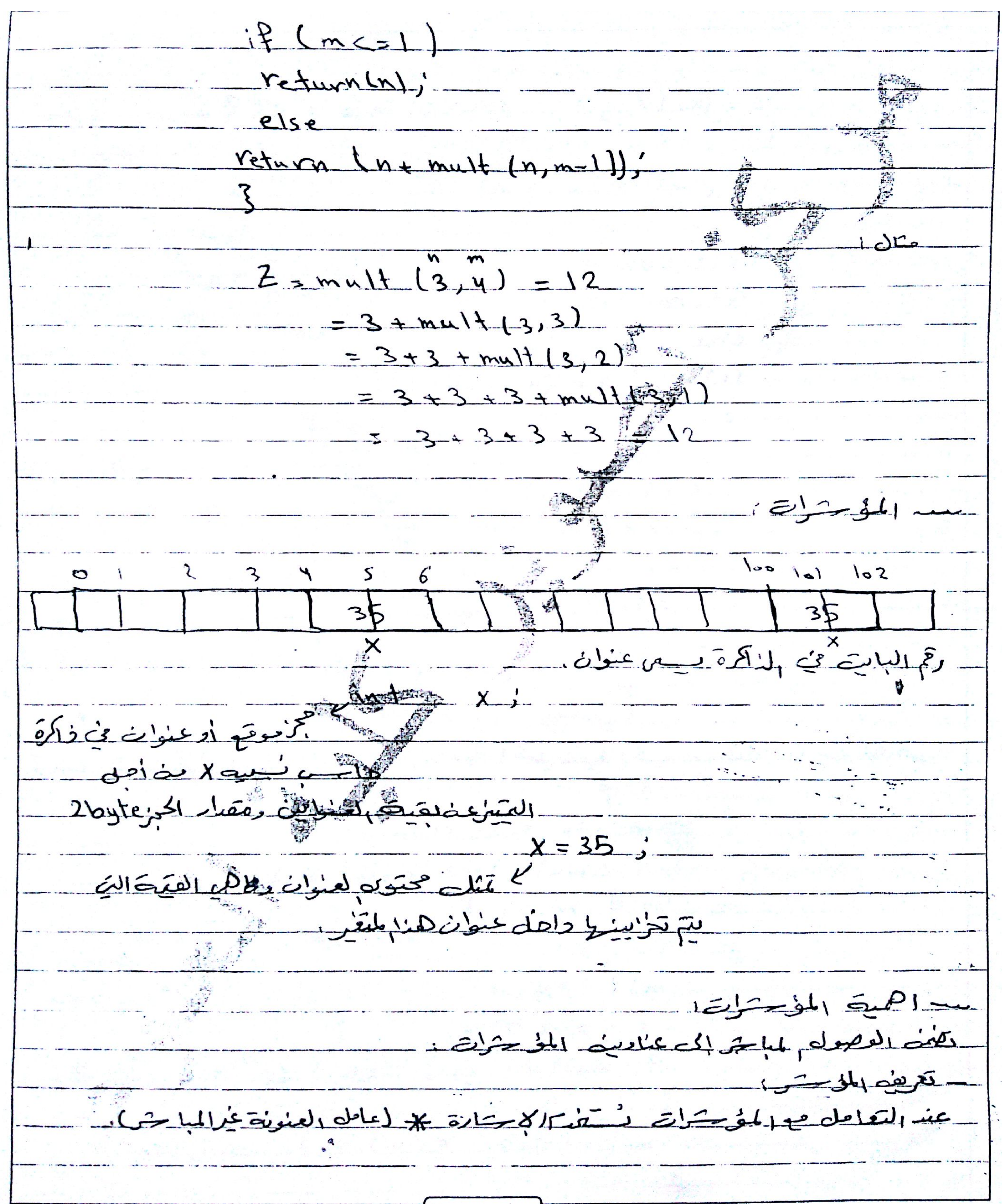
التنفيذ :

a^b

```
.Z = pw(2,3)
 = 2 x pw(2,2)
 = 2 x 2 x pw(2,1)
 = 2x2x2 x pw(2,0) = 8
```

تمرين: باستخدام دوال الإعادة الذاتية اكتب برنامج لحساب جداء عددين صحيحين:

```
#include <iostream.h>
int mult (int n,int m);
void main( )
{
int x,y,z ;
cin>>x>>y;
z=mult (x, y); cout <<"\n z ="<<z;
}
int mult (int n, int m)
{
```



```
if (m<=1)
  return(n);
  else
return (n + mult (n,m-1));
}
```

مثال:

$Z = mult(\overset{n}{3}, \overset{m}{4}) = 12$

$= 3 + mult(3,3)$

$= 3 + 3 + mult(3,2)$

$= 3 + 3 + 3 + mult(3,1)$

$= 3 + 3 + 3 + 3 = 12$

- المؤشرات:

0	1	2	3	4	5	6	...	100	101	102
					35				35	
					x				x	

رقم البايت في الذاكرة يسمى عنوان.

int x;

حجز موقع أو عنوان في ذاكرة الحاسب نسميه x من أجل التعبير عنه بقيمة العنوان ومقدار الحجز 2byte

x = 35;

تمثل محتوى العنوان والقيمة التي يتم تخزينها داخل عنوان هذا المتغير.

- أهمية المؤشرات:

تمكن الوصول المباشر إلى عناوين المؤشرات.

- تعريف المؤشر:

عند التعامل مع المؤشرات نستخدم الإشارة * (عامل العنونة غير المباشر).

صيغته:

; اسم المؤشر * نوع المتغيرات

مثال:

```
int * iptr ;
```

(عنوان)

iptr: المتغير يؤشر إلى عنوان (يؤشر إلى نوع المتغير مخزن في الذاكرة يدل إلى int أي مؤشر يؤشر إلى int)

*iptr: قيمة محتوى العنوان.

مثال:

```
float  * fptr ;
char   * cptr ;
```

كيفية استخدام:

المؤشرات في البرنامج نوضحه عن طريق المثال الآتي.

```
Void main ( )
{
int x, * px ;
  x = 74 ;
px = & x        ← مؤشر يدل إلى عنوان متغير مخزن بالذاكرة.
*px = * & x ;
       =1
cout << x ;        74
cout << * px ;     74   ← طباعة قيمة محتوى العنوان
```

x
74

y

z

المؤشرات:

في الواقع يمكن تمثيل ذاكرة الحاسب على شكل شريط مقسم إلى بايتات وكل بايت في ذاكرة له رقم

رقم البايت يسمى عنوان في الذاكرة.

لابد من التفرقة بين رقم البايت ومحتوى العنوان وهو القيمة التي يتم تخزينها داخل عنوان المتغير .

الصيغة المؤشرات :

في الواقع يمكن كتابة البرنامج من دون المؤشرات ويمكن كتابتها بالإمكان استخدام المؤشرات بهدف الوصول المباشر إلى عناوين المؤشرات .

تعريف المؤشر :

عند تعريف المؤشر تستخدم الإشارة * وتشير على العنونة غير المباشرة والصيغة

نوع المتغيرات ـــ * اسم المؤشر ؛

مثال :

```
int  * iptr ;
```

وهنا لابد من التفريق بين iptr و * iptr

iptr : هو عبارة عن مؤشر يؤشر إلى int (مؤشر = متغير).

* iptr : تمثل محتوى العنوان وهي القيمة التي يتم تخزينها داخل عنوان المتغير

مثال :

```
Float  * Fptr ;
char  * cptr ;
```

تمرين

باستخدام المؤشرات اكتب برنامج جمع عددين صحيحين

```
#include <iostream.h>
void  main( )
{
int  x , y , * px , * py , z ;
  x = 10 ;
  y = 35 ;
  px = & x ;      عنوان x
  py = & y ;      عنوان y
  z = * px  +  * py ;
         فراغات
  cout << z ;
```

}

ملاحظة:
عند استخدام الدوال في البرنامج يمكن للبارامترات أن تكون مؤشرات بالشكل:
عند تعريف الدالة التي بارامتراتها مؤشرات نسبق البارامترات بـ *
وأما عند استدعاء الدالة نسبقها بإشارة العنوان (إشارة العنوان قبل البارامتر):

مثال:
```
int = sum(int *a, int *b);
c= sum(&a , &b);
```

تمرين: باستخدام مفهوم الدالة والمؤشر اكتب برنامج جمع عددين صحيحين:

```
#include <iostream.h>
int sum( int *a, int *b);
Void main()
{
int x,y,s;
cin>>x>>y;
s=sum(&x , &y);
cout <<"\n s="<<s;
}
int sum (int *a, int *b)
{
int c;
c= *a + *b;
return(c);
}
```

ملاحظة:
يمكننا في الواقع تمرير البارامترات إلى البرنامج الرئيسي عن طريق:
١- تمرير البارامترات بالقيمة:

٢- تمرير البارامترات بالمؤشر .

٣- تمرير البارامترات بالعنوان :

تمرين ١: اكتب برنامج بسيط يحسب مجموع عددين صحيحين عن طريق تمرير البارامتر بالعنوان .

```
#include <iostream.h>
Void sum(int a, int b, int &c);
Void main
{
Int x,y,z;
cin>>x>>y;
sum(x,y,s);
cout<<"\n s = "<<s;
{
Void sum(int a,int b,int &c)
}
c=a+b;
}
```

البحث الثنائي : (خوارزمية البحث الثنائي) :

دليل	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
عناصر المصفوفة	0	2	4	6	8	10	12	14	16	18	20	22	24	26	28

16 18 20 22 24 26 28

24 26 28

24

$\frac{0+14}{2} = 7$

$\frac{8+14}{2} = ١١$, $\frac{12+14}{2} = 13$

key = 25 غير موجود .

اذا كانت لدينا مصفوفة تحتوي على عناصر لنبين فيما اذا كانت المصفوفة تحتوي على قيمة تتطابق قيمتها مع قيمة نبحث عنها تسمى المفتاح key

عملية إيجاد Key ضمن عناصر المصفوفة ، رسم عملية البحث:

وعملية البحث الثنائي تعتمد على حذف نصف عناصر المصفوفة بعد كل مقارنة ففي المرحلة الأولى نقارن العنصر الواقع في منتصف المصفوفة مع العنصر Key إذا تطابق العنصرين يكون العنصر موجود و~~...~~ نطبع دليل المصفوفة وعند عدم التطابق نحذف نصف عدد عناصر المصفوفة وإذا كانت قيمة العنصر Key أكبر من المنتصف نحذف العناصر على اليسار وإلا العناصر على اليسار ثم نكرر عملية المقارنة ونحذف نصف عدد العناصر حتى نحصل على مصفوفة متكونة من عنصر واحد فقط وفي هذه الحالة نقول أن العنصر موجود ونطبع الدليل أو نطبع العنصر غير موجود.

مثال يوضح إيجاد Key

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
0	2	4	6	8	10	12	14	16	18	20	22	24	26	28
0	2	4	6	8	10	12								
				8	10	12								
				8										

موجود:

مثال:

```
#include <iostream.h>
        int bns( int b[] ;int sk, int Low ; int high; int size);
int void main ( )
    {
        Cout   int   ars = 15 ;
        int  a[ars], Key, result, i ;
        For (i=0 ; i< ars ; ++i)
        a[i] = 2*i
        count << "\n Key = "; cin>> Key ;
    result = bns (a, Key, 0, ars-1, ars)
        if (result != -1)
```

نتيجة استدعاء الدالة (result)

عناصر المصفوفة ← a[i] = 2*i

استدعاء الدالة ← result = bns (...)

```
cout <<"\n found " << result;
else
cout <<" Not found ";
return(0);
}
```

شرح البرنامج

```
int bns (int b[ ], int sk, int Low, int high, int size)
```

sk: عنصر مراد البحث عنه — Low: دليل الأول — high: دليل الأخير

```
{
int m;
while (Low <= high)
{
m=(Low + high)/2;
if (b[m] == sk)
  return(m);
  else
if (sk < b[m])
    high=m-1;
  else
  Low=m+1;
}
return(-1);
}
```

انتهت المحاضرة